To appear in *Advanced Robotics*
Vol. 30, No. 21, Nov. 2016, 1380–94

FULL PAPER

Iterative Path-Accurate Trajectory Generation for Fast Sensor-Based Motion of Robot Arms

Friedrich Lange[a*] and Alin Albu-Schäffer[ab]

[a]*Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany;*

[b]*Department of Informatics, Technical University Munich, Garching, Germany*

(received October 2015, revised March 2016, June 2016, accepted July 2016, online August 2016)

Sensor-based trajectory generation of industrial robots can be seen as the task of, first, adaptation of a given robot program according to the actually sensed world, and second, its modification that complies with robot constraints regarding its velocity, acceleration, and jerk. The second task is investigated in this paper. Whenever the sensed trajectory violates a constraint, a transient trajectory is computed that, both, keeps the sensed path, and reaches the sensed trajectory as fast as possible while satisfying the constraints. This is done by an iteration of forward scaling and backtracking. In contrast to previous papers a new backtracking algorithm and an adaptation of the prediction length are presented that are favorable for high-speed trajectories. Arc Length Interpolation is used in order to improve the path accuracy. This is completed by provisions against cutting short corners or omitting of loops in the given path. The refined trajectory is computed within a single sampling step of 4 ms using a standard KUKA industrial robot.

Keywords: Trajectory Generation, Industrial Robots, Motion Generation, Path Accuracy

1. Introduction

1.1. *Problem Formulation*

In this paper, trajectory generation is considered for sensor-based motion of robot arms. We assume a *reference trajectory* $\mathbf{x}_r(k+\kappa)$, which at time step k is defined for the current and several future sampling steps $k+\kappa$ with $\kappa \geq 0$, together with the desired sensor values $\mathbf{s}_d(k+\kappa)$. Whenever the measured sensor values $\mathbf{s}(k)$ do not coincide with the desired values, the reference trajectory is converted online to a *sensed* or *desired* trajectory $\mathbf{x}_d(k+\kappa)$ in order to adapt to a changed environment, as in [1]. However, this sensed trajectory might violate constraints of the robot that are given by the robot manufacturer. Therefore, the trajectory undergoes an additional adaptation step, where the result is denoted as *commanded* trajectory, whose current position $\mathbf{x}_c(k)$, or its representation in the axis space $\mathbf{q}_c(k)$, is sent to the servo controller that is designed by the robot manufacturer (see Fig. 1).

The conversion from the sensed trajectory to the commands is denoted as *trajectory generation*. The conversion is fundamental since industrial controllers do not tolerate any exceeding of the given constraints and, in that case, execute an emergency stop. The constraints are usually ignored, e.g. in [2, 3], which is tolerable in the presence of sufficiently small control gains that are computed in a control design aiming at smooth signals. Previous sensor-based control methods of the authors [1, 4] used simpler methods for trajectory generation, which is not suitable for fast motion at high sampling rates. Then a more refined trajectory generation method has to be chosen.

*Corresponding author. Email: Friedrich.Lange@dlr.de

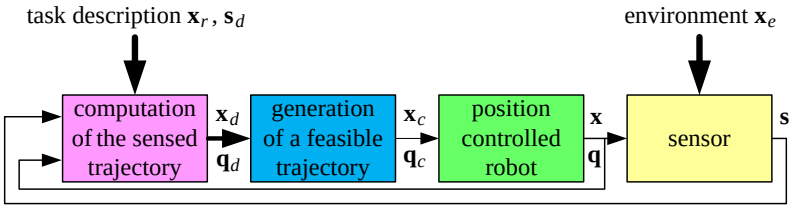

Figure 1. Sensor-based control

In addition to constraints on the position, the velocity, and the acceleration, we assume that the jerk, i.e.,the rate of changes of the acceleration, is limited. The consideration of the jerk is preferable for limited voltage of the motors, reduced wear on the robot, and minimum excitation of oscillations [5, 6]. It is therefore present in today's industrial robot controllers. A trajectory that complies with these constraints is called *feasible*.

In most papers a feasible trajectory is computed in such a way that the execution of the resulting trajectory takes minimal time. Alternatively, a trajectory is generated that is feasible without a reduction of the programmed speed, as in [7]. Instead, our goal is to keep the geometrical *path* of the sensed trajectory where *path* comprises position and orientation. Depending on the application, *path accuracy* is fundamental for performance and for safety reasons.

It should be noted that, in contrast to the offline generation of a whole trajectory, it is not always possible to generate a feasible, path-accurate trajectory from a given state. For example, it may be impossible to inhibit overshooting. In contrast to acceleration phases, the path-accuracy is at risk whenever the desired *deceleration* exceeds the constraints, even more when some axes are decelerated whereas others are accelerated. This is discussed in more detail in [8, 9].

We consider the accuracy of the *commanded* path, not of the actually executed path. The difference owing to the robot dynamics is not yet implemented. Thus we do not need to know the robot dynamics nor the industrial controller.

As a further criterion besides feasibility and path accuracy, the commanded trajectory hangs the sensed motion as fast as possible, i.e., it is synchronized with it. The sensed trajectory, on its part, will have only small temporal deviations from the reference trajectory. It is crucial to keep the timeline instead of the time-optimal motion of a single robot, at least for applications with multiple robots or conveyor belts.

Challenging applications of trajectory generation feature fast motion with high accuracy requirements, e.g. when approaching an object, where the sensed path abruptly changes when a contact force is measured. In such cases, in order to avoid excessive forces as well as a loss of contact after the first overshooting, the robot should decelerate the normal motion as fast as possible while performing the desired tangential motion, e.g. polishing. Another critical task for trajectory generation is stopping during constrained motion without exerting undesired forces, i.e., without leaving the previously defined path. This has been investigated in [10].

Sudden changes of the desired trajectory can also be present before sensing a contact, e.g. in pick-and-place operations or during assembly, when the exact gripping pose is not known until it is sensed (e.g. by an eye-in-hand camera of limited field of view). A significant change of orientation may then be needed in a very short time. This path correction should be executed without colliding with the object to be gripped, i.e., without deviations from the sensed path.

1.2. *Previous Work*

There is a lot of previous work on trajectory generation, beginning in the eighties, e.g. [11, 12]. Nowadays the Reflexxes Motion Library [8, 13, 14] is widely used. It provides a useful trajectory

generator that considers the full current state. A new trajectory can be computed on-the-fly, i.e., without stopping the robot. However, the goal is only to reach a target pose with a given velocity and acceleration. The executed path is usually not considered, besides a straight line [15].

In order to keep a desired path, Refs. [5, 16–22] generate a trajectory through several points, usually by polynomial interpolation or by splines. However, it is not assumed that the given points represent all sampled positions. A closer following of a desired path is achieved by [6, 9, 12, 23–35]). Ref. [36] combines trajectory generation and feedback control in order to reduce the deviations from the desired path. Mostly, the execution time is minimized (Time-Optimal Path Parameterization or TOPP). Synchronization with a given trajectory, as requested here, is the goal only in [26, 27, 32].

Trajectory generation along a desired path is also denoted as generating a velocity profile for the given path. The path is typically represented with the arc length s as a parameter. The task is to compute the scalar velocity $\dot{s}(s)$ in the phase space that satisfies the constraints. Early papers as [12, 24, 25] did not consider constraints on the jerk. Ref. [6] uses forward and backward computation and smoothing at the meeting point. Ref. [9] only computes from the beginning of the trajectory, but switches before entering a *trapped area*, since otherwise the path will be lost later on. Ref. [27] locally restricts the given velocity limits in order to satisfy the constraints on the acceleration and the jerk. Ref. [29] assumes 4^{th} order functions for s. Ref. [33] defines a maximum velocity curve that, besides the other constraints, serves as an upper limit for $\dot{s}$.

Unlike the other methods, Ref. [35] does not convert the kinematic constraints or the shape of the path to constraints on s, which then result in a velocity profile. Instead, it is proposed to predict future constraints and in doing so to modify the trajectory at the preceding time steps. The computation does not assume a differentiable desired trajectory. The strategy will be summarized in Sect. 2. In contrast to [35], here the view is generic, i.e., independent from the type of interpolation.

The motion is predominantly divided into a limited number of phases with given characteristics as a constant jerk or a constant acceleration (see e.g. [6, 8, 9, 25, 32]). New phases are introduced depending on the path where the task is to find the appropriate switching points [6, 9, 25], if the number of phases is not given a priori. The latter implies assumptions about the shape of the desired path, e.g. linear or circular [32].

In addition to the above task requirements, Refs. [9, 37–39] allow to optimize further degrees of freedom (dof). In this way, collisions of the robot arm may be inhibited while tracking the pose of the tool center point (tcp). However, such a computation is not applicable for usual 6-axis industrial robots.

1.3. *Organization of the Paper*

Sect. 2 explains the proposed iterative trajectory generation. Sect. 3 then reviews the Arc Length Interpolation (ALI) from [35]. New aspects for improving the performance are presented in Sect. 4. This includes a new backtracking algorithm (Sect. 4.1), an adaptation of the prediction length (Sect. 4.2), and provisions against cutting short corners or omitting of loops in the given path (Sect. 4.3). Their contributions, especially for high speed trajectories, are demonstrated in experiments with a KUKA industrial robot in Sect. 6. Before, performance, stability and convergence of the method are discussed in Sect. 5. Finally the paper is concluded in Sect. 7.

2. Iterative Forward Scaling and Backtracking

The following considerations are formulated in the axis space (joint space) of the robot, since the constraints are defined there. Vectors of all axes are denoted by bold face letters whereas single axes are in normal face with the index as last subscript, e.g. $\mathbf{q} = (q_1, \cdots, q_6)^T$. Without loss of generality, in this paper we omit the sampling time by assuming $T_0 = 1$ which means that the time is expressed in steps instead of in seconds. The velocity $\mathbf{v}$, the acceleration $\mathbf{a}$, and the jerk $\mathbf{j}$

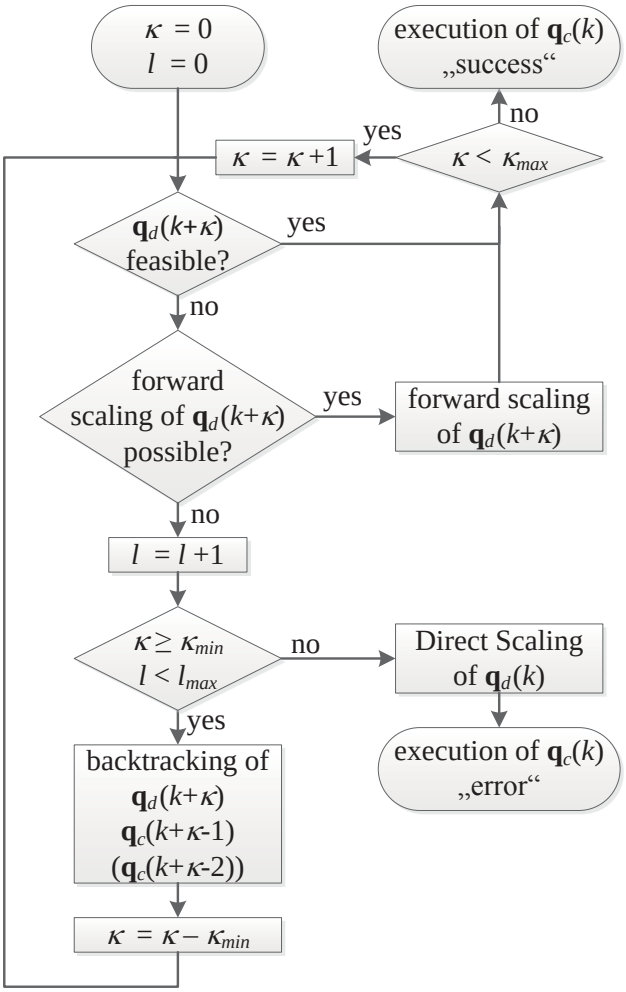

Figure 2. Simplified flow chart. Scaling and backtracking only apply for the constraints that are otherwise violated. Accordingly κ_{min} is 1 or 2 for violations on the acceleration limit or the jerk limit, respectively.

are expressed by backward differences. This results in the notation shown in Appendix A, similar to [32]. The respective limits of an axis i are denoted as $\pm\bar{v}_i$, $\pm\bar{a}_i$, and $\pm\bar{j}_i$, and may be time dependent as the signals themselves.

The current command $\mathbf{q}_c(k)$ that is sent to the servo controller coincides with the position $\mathbf{q}_d(k)$ of the sensed trajectory as long as the sensed trajectory is feasible. If any of the constraints

$$|v_{di}(k)| = |q_{di}(k) - q_{ci}(k-1)| \leq \bar{v}_i \quad (1)$$

$$|a_{di}(k)| = |q_{di}(k) - 2q_{ci}(k-1) + q_{ci}(k-2)| \leq \bar{a}_i \quad (2)$$

$$|j_{di}(k)| = |q_{di}(k) - 3q_{ci}(k-1) + 3q_{ci}(k-2) - q_{ci}(k-3)| \leq \bar{j}_i \quad (3)$$

is violated, the following subsections have to be considered, replacing the sensed positions $q_{di}(k)$ by commanded positions $q_{ci}(k)$ that satisfy the constraints, as in Fig. 2. If an iteration is required, preliminary $q_{ci}(k)$ will be renamed by $q_{di}(k)$ in order to serve as input for the next iteration step.[1]

[1]In an analogous manner to (1)-(3) the differences of the q_{ci} are denoted by v_{ci}, a_{ci}, and j_{ci}.

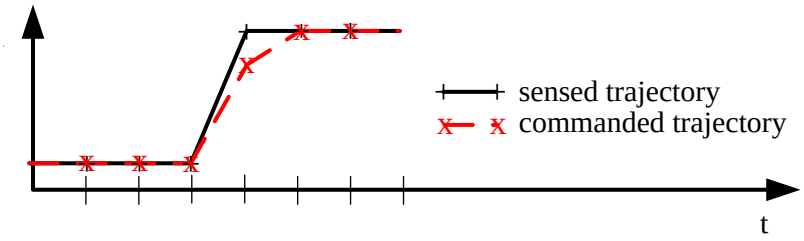

Figure 3. Modification of a 1 dof path by scaling the velocity at the discontinuity of the sensed path in order to satisfy a velocity constraint.

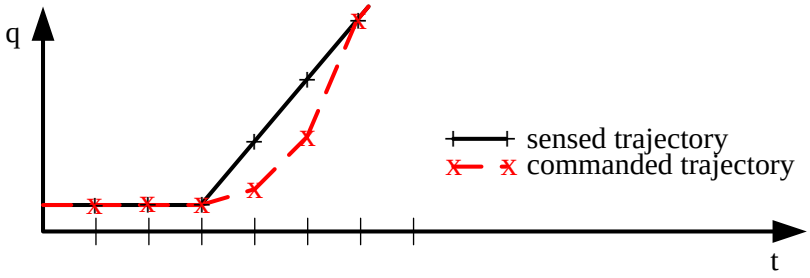

Figure 4. Modification of a 1 dof path by scaling the velocity at the corner of the sensed path in order to satisfy an acceleration constraint.

2.1. *Forward Scaling of the Velocity*

If a constraint of axis i is violated, the first option is to scale the velocity $v_{di}(k)$ of this axis, since in this way the velocity, the acceleration, and the jerk may be reduced. Figs. 3 and 4 show simple examples. Equal scaling of the velocities of all axes was considered first, as in [40], since it seems not to compromise the path accuracy. Strictly speaking, a suitable implementation of this *forward scaling* has to be found that modifies the positions of all axes such that path accuracy is maintained. This is explained in Sect. 3.

However, it is not always possible to reduce an acceleration $a_{di}(k)$ or a jerk $j_{di}(k)$ by reducing the velocity $v_{di}(k)$. For example, if the desired acceleration is against the direction of the velocity, i.e., if the robot is decelerated, a reduction of the velocity will increase the absolute value of the acceleration. Therefore, in order to get a feasible trajectory, i.e., a $q_{ci}(k)$ that satisfies the acceleration limit, the desired velocity cannot be further reduced. Thus $|v_{ci}(k)| < |v_{ci}(k-1)|$ can be reached, but not $|v_{ci}(k)| \leq |v_{di}(k)|$, i.e., the robot will overshoot by exceeding $q_{di}(k)$. This may result in an oscillation around the sensed position.

Furthermore, it can be noted that a reduction of the velocity that is requested by an axis i_1 may result in an intolerable deceleration for another axis i_2 that, for its part, would be feasible without the scaling. Therefore, a violation of the constraints may exist not only during deceleration.

2.2. *Backtracking from Predictions*

Overshooting or other problems of forward scaling may be inhibited by predictively checking the constraints, i.e., $|v_{di}(k+\kappa)| \leq \bar{v}_i$, $|a_{di}(k+\kappa)| \leq \bar{a}_i$, and $|j_{di}(k+\kappa)| \leq \bar{j}_i$, for $0 \leq \kappa \leq \kappa_{max}$ with a prediction length κ_{max} that is sufficient. Its adaptation will be investigated in Sect. 4.2.

Note that the signals are defined as the differences between the sensed, desired positions $q_{di}(k+\kappa)$ and the preceding executed positions $q_{ci}(k+\kappa-\cdots)$, as in (1)-(3), since the latter may already be scaled by a preceding step.

Whenever a constraint cannot be resolved by forward scaling, the error will be processed by *backtracking* in order to inhibit the violation. A violated constraint on $j_{di}(k+\kappa)$ will result in a

modification of $q_{di}(k+\kappa)$, $q_{ci}(k+\kappa-1)$, and $q_{ci}(k+\kappa-2)$. On the other hand, if $|a_{di}(k+\kappa)|$ exceeds the limit $\bar{a}_i$, only $q_{di}(k+\kappa)$ and $q_{ci}(k+\kappa-1)$ are modified. A violation of the velocity limit $\bar{v}_i$ does not need backtracking since it can always be resolved by forward scaling. A suitable implementation of this type of backtracking will be explained in Sect. 3. It always generates a feasible $q_{ci}(k+\kappa)$, but not necessarily a feasible $q_{ci}(k+\kappa-1)$ or $q_{ci}(k+\kappa-2)$.

After backtracking, the computed $q_{ci}(k+\kappa)$ to $q_{ci}(k+\kappa-\kappa_{min})$ are copied to q_{di}, and forward scaling is continued from the first modified position, i.e., $q_{di}(k+\kappa-\kappa_{min})$, with $\kappa_{min} = 1$ or 2 for acceleration or jerk constraints, respectively. If, perhaps after forward scaling at time step $k+\kappa-\kappa_{min}$, a signal $a_{ci}(k+\kappa-\kappa_{min})$ or $j_{ci}(k+\kappa-\kappa_{min})$ does not satisfy the constraint, further backtracking is immediately required. Otherwise forward scaling is performed for the next time step in order to get a feasible $q_{ci}(k+\kappa-\kappa_{min}+1)$. In this way the axis positions are iteratively modified by forward scaling and backtracking until a feasible trajectory is found for all time steps $k \le k+\kappa \le k+\kappa_{max}$. Fig. 2 summarizes the procedure.

2.3. *Direct Scaling (DS)*

There are two reasons to abort the iteration before the solution is found:

- If the backtracking reaches $\kappa = 0$, the modification of $q_{ci}(k+\kappa)\cdots q_{ci}(k+\kappa-\kappa_{min})$ at time step k is not possible anymore.
  - For $\kappa_{min} = 2$ the backtracking is impossible even for $\kappa = 1$. A modified backtracking might then help. In most cases this does not result in a feasible trajectory.
- Since no upper limit for the number of iteration steps can be given, it is possible that the available computing time is exceeded. This is prevented by stopping the iteration whenever the number of iteration steps l reaches a threshold l_{max}. Then there are two cases:
  - The so far computed $q_{ci}(k)$ is feasible. Then it can be executed. However, it may be impossible to maintain the given path in a later time-step.
  - The so far computed $q_{ci}(k)$ is not feasible.

Whenever $q_{ci}(k)$ is not feasible, still there is a way to get a feasible trajectory by *Direct Scaling* (DS), but the trajectory is not path accurate anymore.

Direct Scaling directly reduces the constrained signal by $\mathbf{a}_c(k) = \alpha_a \mathbf{a}_d(k)$ or

$$\mathbf{q}_c(k) = \mathbf{q}_c(k-1) + \mathbf{v}_c(k-1) + \alpha_a \mathbf{a}_d(k) \tag{4}$$

and by $\mathbf{j}_c(k) = \alpha_j \mathbf{j}_d(k)$ or

$$\mathbf{q}_c(k) = \mathbf{q}_c(k-1) + \mathbf{v}_c(k-1) + \mathbf{a}_c(k-1) + \alpha_j \mathbf{j}_d(k), \tag{5}$$

as explained in Appendix A.

In (4) and (5) all axis accelerations or jerks are reduced by the same factor α_a or α_j respectively. These factors are computed as the maximum values that satisfy the constraints, i.e.,

$$\alpha_a = \min_{i,\ |a_{di}(k)|>\bar{a}_i} (\bar{a}_i / |a_{di}(k)|) \tag{6}$$

and

$$\alpha_j = \min_{i,\ |j_{di}(k)|>\bar{j}_i} (\bar{j}_i / |j_{di}(k)|). \tag{7}$$

This results in $0 < \alpha_* < 1$ with $* \in \{a, j\}$, if the constraints are violated before. Thus Direct Scaling always gives a feasible commanded trajectory. Fig. 5 shows the effect of Direct Scaling of

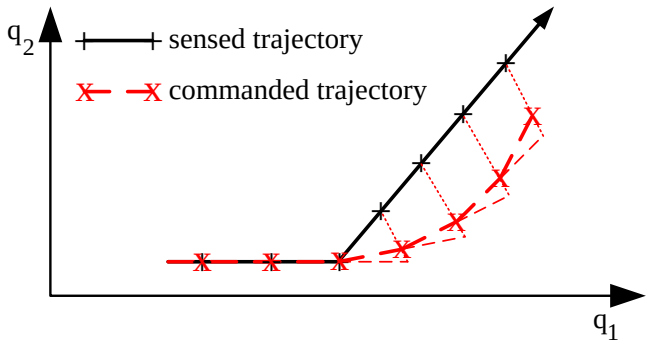

Figure 5. Modification of a 2 dof path by Direct Scaling of the acceleration at the corner of the sensed path in order to satisfy the acceleration constraints. Further steps might be found by forward scaling.

the acceleration. The computed positions $\mathbf{q}_c(k)$ are between the sensed, desired positions $\mathbf{q}_d(k)$ that cannot be reached and the positions that would result from zero acceleration, i.e., $\mathbf{q}_c(k-1)+\mathbf{v}_c(k-1)$. Direct Scaling of the jerk results in a position between $\mathbf{q}_d(k)$ and the position which would be reached with zero jerk, i.e., $\mathbf{q}_c(k-1)+\mathbf{v}_c(k-1)+\mathbf{a}_c(k-1)$.

Direct Scaling is typically required whenever the sensed trajectory conflicts with physical limits, e.g. when the sensed trajectory is recomputed because of unexpected sensor data and this new trajectory implies a sudden stop or even a step back at the current time step. An overshooting is then unavoidable. This has been shown in [35] where a force-sensor detects an unexpected contact during high speed motion. The chance of a path-accurate solution that meets all requirements increases with the span of time for which big accelerations or jerks can be predicted.

There are further cases in which the normal backtracking fails. For example, an oscillating sensed trajectory may require too many iteration steps. Furthermore, once the velocity is reduced too much, it will not be increased again. This is the motivation for the refined backtracking that will be presented in Sect. 4.1.

3. Arc Length Interpolation (ALI)

An interpolation scheme is required for the implementation of both, forward scaling according to Sect. 2.1 and backtracking according to Sect. 2.2. Instead of the direct position interpolation (DPI) used in [40], we now prefer the *Arc Length Interpolation* (ALI) as presented in [35].

3.1. *Interpolation for Forward Scaling*

Whenever a constraint (1)-(3) is violated, the constrained component is set to the limit, i.e.,

$$q_{fvi}(k) = q_{ci}(k-1) \pm \bar{v}_i \quad (8)$$

$$q_{fai}(k) = q_{ci}(k-1) + v_{ci}(k-1) \pm \bar{a}_i \quad (9)$$

$$q_{fji}(k) = q_{ci}(k-1) + v_{ci}(k-1) + a_{ci}(k-1) \pm \bar{j}_i, \quad (10)$$

where the sign of the limit is such that the modification with respect to $q_{di}(k)$ is minimum.

Then, in order to preserve the shape of the sensed path, for each $q_{f*i}(k)$ with $* \in \{v, a, j\}$ a scalar parameter s_{*i} is computed which, roughly speaking, represents the traveled path. This parameter is usually called the *arc length*. Here it is defined by the time parameter of the original trajectory, i.e., $s_{*i} = k$ points to the position $\mathbf{q}_d(k)$.

$s_{*i}(k)$ is computed from $q_{f*i}(k)$ for $\underline{s}_{*i}(k) = k, k-1, \cdots, \underline{s}(k-1)+1$ by

$$s_{*i}(k) = \underline{s}_{*i}(k) + \frac{q_{f*i}(k) - q_{di}(\underline{s}_{*i}(k))}{q_{di}(\underline{s}_{*i}(k)+1) - q_{di}(\underline{s}_{*i}(k))}, \tag{11}$$

where $\underline{s}(k-1)$ is the biggest integer that is not greater than $s(k-1)$. Computing (11) with the different values of $\underline{s}_{*i}(k)$ is stopped whenever an $s_{*i}(k)$ fits to the interval

$$\underline{s}_{*i}(k) \leq s_{*i}(k) < \underline{s}_{*i}(k) + 1. \tag{12}$$

If in this way no solution is found,

$$s_{*i}(k) = s(k-1) + \frac{(q_{f*i}(k) - q_{ci}(k-1)) \cdot (\underline{s}(k-1)+1-s(k-1))}{q_{di}(\underline{s}(k-1)+1) - q_{ci}(k-1)} \tag{13}$$

is computed. Then it is checked whether $s_{*i}(k)$ satisfies

$$s(k-1) \leq s_{*i}(k) < \underline{s}(k-1) + 1. \tag{14}$$

Otherwise backtracking or Direct Scaling is required.

The smallest value $s_{*i}(k)$ of all axes and constraints is taken as $s(k)$, meaning that the most restricting constraint has to be satisfied. If no constraint (1)-(3) is violated, $s(k)$ is not modified from its initial value of $s(k) = k$.

$\mathbf{q}_c(k)$ can be computed from $s(k)$ by

$$\mathbf{q}_c(k) = \mathbf{q}_d(\underline{s}(k)) + (s(k) - \underline{s}(k)) \cdot (\mathbf{q}_d(\underline{s}(k)+1) - \mathbf{q}_d(\underline{s}(k))), \tag{15}$$

or by

$$\mathbf{q}_c(k) = \mathbf{q}_c(k-1) + \frac{(s(k) - s(k-1)) \cdot (\mathbf{q}_d(\underline{s}(k-1)+1) - \mathbf{q}_c(k-1))}{\underline{s}(k-1) + 1 - s(k-1)}, \tag{16}$$

if $s(k) < \underline{s}(k-1)+1$. This results in $q_c(k) = q_{f*i}(k)$ for the most limiting constraint.

Then the constraints (1)-(3) are checked with $q_c(k)$ instead of $q_d(k)$ since the modification from $q_d(k)$ to $q_c(k)$ may have affected them. The procedure is repeated, if a constraint is violated, disregarding all results with $s_{*i}(k) > s(k)$ where $s(k)$ is the result of the previous iteration step. A feasible solution is found whenever all constraints are satisfied. The sequence is summarized in Fig. 6.

When forward scaling is used after backtracking, $s(k)$ is not initialized by k, but (11) is first executed with $\underline{s}_{*i}(k) = \underline{s}(k)$, with $s(k)$ from a previous step of the main iteration.

Fig. 7 shows an example of a curved trajectory that, because of the acceleration constraints, cannot be executed as sensed. The generated trajectory leaves the sensed path when computed with DPI, whereas it exactly tracks the path when ALI is used. Jerk constraints are not considered in this example.

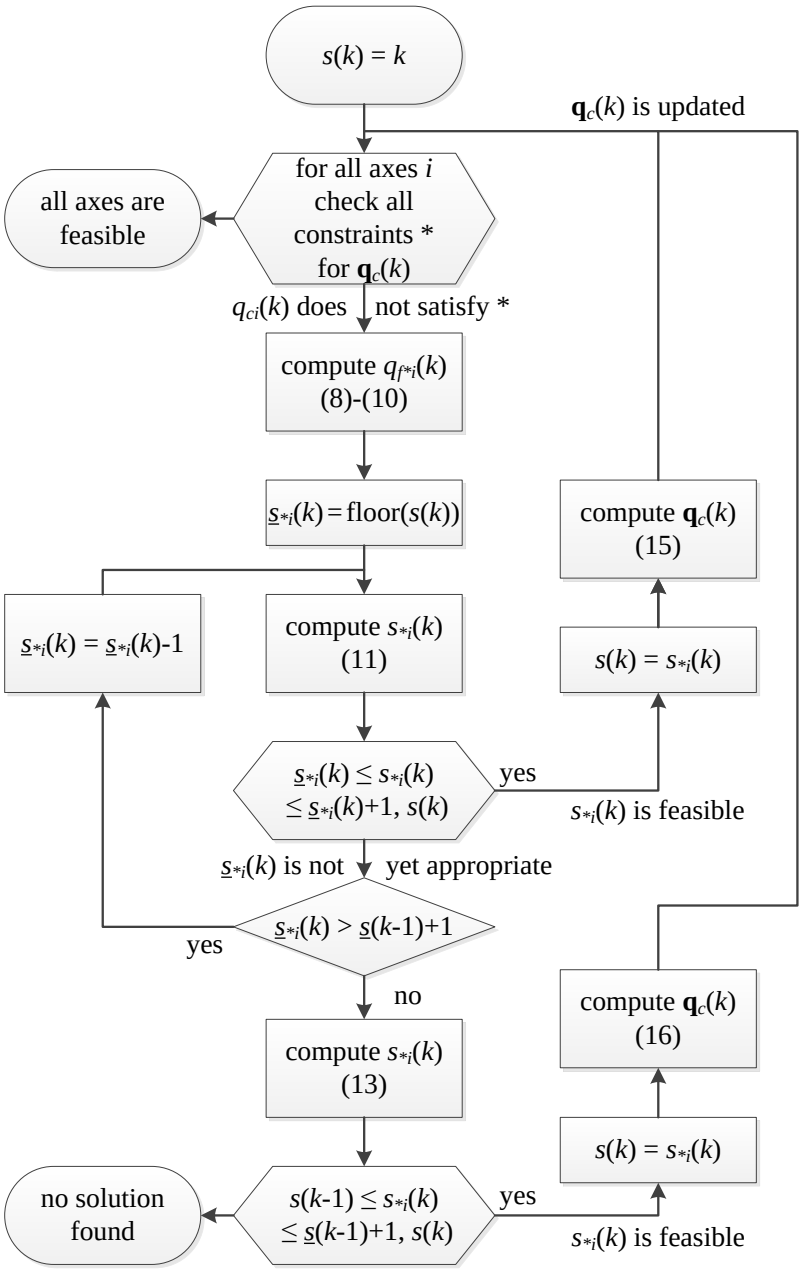

Figure 6. Flow chart of ALI during forward scaling.

3.2. *Interpolation for Backtracking*

For brevity, in the sequel we denote $k + \kappa$ as k'. If the acceleration limit cannot be resolved by forward scaling, we use

$$q_{bai}(k') = q_{ci}(k'-1) + \alpha_{a_i}(q_{di}(k') - q_{ci}(k'-1)) \quad (17)$$

$$q_{bai}(k'-1) = q_{ci}(k'-2) + (1+\alpha_{a_i})(q_{ci}(k'-1) - q_{ci}(k'-2))/2, \quad (18)$$

where $(1 + \alpha_{a_i})/2$ is selected as scaling factor for (18) such that

$$\alpha_{a_i} = \bar{a}_i/|a_{di}(k)| \quad (19)$$

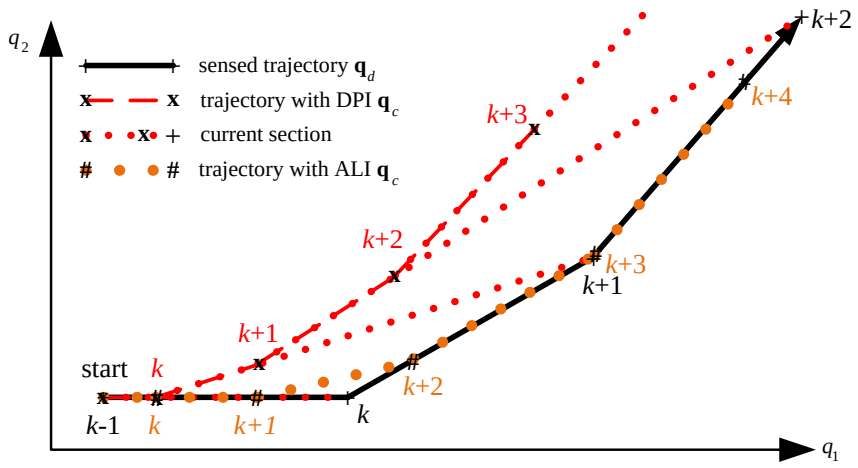

Figure 7. Example with forward scaling along a curved sensed path (black): The robot is started in time step $k-1$. Then $\mathbf{q}_c(k)$ is computed because $\mathbf{q}_d(k)$ violates the acceleration constraint. This is identical with DPI and ALI. In the next step, scaling the line from $\mathbf{q}_c(k)$ to $\mathbf{q}_d(k+1)$ (DPI) computes the red $\mathbf{q}_c(k+1)$. Then scaling between $\mathbf{q}_c(k+1)$ and $\mathbf{q}_d(k+2)$ results in the red $\mathbf{q}_c(k+2)$ (DPI). Instead, ALI computes the orange $\mathbf{q}_c(k+1)$ from (13) and (16), i.e., between the previously executed position $\mathbf{q}_c(k)$ and the next sensed position $\mathbf{q}_d(k)$ (not $\mathbf{q}_d(k+1)$). This differs from the computation of $\mathbf{q}_c(k+2)$ in (11) and (15), i.e., between two sensed positions $\mathbf{q}_d(k)$ and $\mathbf{q}_d(k+1)$ (without considering $\mathbf{q}_d(k+2)$). With ALI there is no path error at the sampled positions $\mathbf{q}_c$.

gives $|a_{bai}(k')| = \bar{a}_i$ for the most constrained axis, whereas $\alpha_{a_i} = 0$ results in $a_{bai}(k') = 0$.[2]

The effect of different values of α_{a_i} in (17)-(18) is illustrated in Figs. 8 and 9. Fig. 8 also shows that $\mathbf{a}_c(k'-1)$ may still be nonzero even after a modification of $\mathbf{q}_c(k')$ and $\mathbf{q}_c(k'-1)$ with $\alpha_{a_i} = 0$. Fig. 9 shows that each step of backtracking with $0 < \alpha_{a_i} < 1$ smoothes corners of the sensed path.

The approach for a violation of the jerk limit is

$$\begin{aligned} q_{bji}(k') &= q_{ci}(k'-1) + \alpha_{j_i}(q_{di}(k') - q_{ci}(k'-1)) && (20)\\ q_{bji}(k'-1) &= q_{ci}(k'-2) + (1+2\alpha_{j_i})(q_{ci}(k'-1) - q_{ci}(k'-2))/3 && (21)\\ q_{bji}(k'-2) &= q_{ci}(k'-3) + (2+\alpha_{j_i})(q_{ci}(k'-2) - q_{ci}(k'-3))/3, && (22) \end{aligned}$$

where the factors $(1+2\alpha_{j_i})/3$ and $(2+\alpha_{j_i})/3$ are chosen because

$$\alpha_{j_i} = \bar{j}_i / |j_{di}(k)| \qquad (23)$$

results in $|j_{bji}(k')| = \bar{j}_i$ for the most constrained axis.

Fig. 10 illustrates backtracking of the jerk. As a side effect the accelerations $a_c(k')$ and $a_c(k'-1)$ are reduced. With $\alpha_{j_i} = 0$ the acceleration $a_c(k') = a_c(k'-1) = 1/3\ a_d(k'-1)$ is constant.

Equations (17)-(18) and (20)-(22) provide a feasible trajectory, but path accuracy is not guaranteed. Therefore, as with forward scaling, the arc length is computed from the $q_{b*i}(k'')$ with $k'' = k', \cdots, k' - \kappa_{min}$ and $* \in \{a, j\}$. Then, instead of $s(k''-1) \le s_{*i}(k'') \le s(k'')$, the conditions $s(k'-2) \le s_{*i}(k'-1) \le s_{*i}(k') \le s(k')$ and $s(k'-3) \le s_{*i}(k'-2) \le s_{*i}(k'-1) \le s_{*i}(k') \le s(k')$ have to be satisfied for (17)-(18) and (20)-(22) respectively. With $k^- = k' - 1 - \kappa_{min}$ this results in

$$s_{*i}(k'') = \underline{s}_{*i}(k'') + \frac{q_{b*i}(k'') - q_{di}(\underline{s}_{*i}(k''))}{q_{di}(\underline{s}_{*i}(k'')+1) - q_{di}(\underline{s}_{*i}(k''))} \qquad (24)$$

[2] $a_{bai}(k')$ and $j_{bji}(k')$ are defined analogously to (2) and (3), by using the just computed $q_{b*i}(k')$ to $q_{b*i}(k' - \kappa_{min})$ and $q_{ci}(k' - \kappa_{min} - 1)$ with $* \in \{a, j\}$.

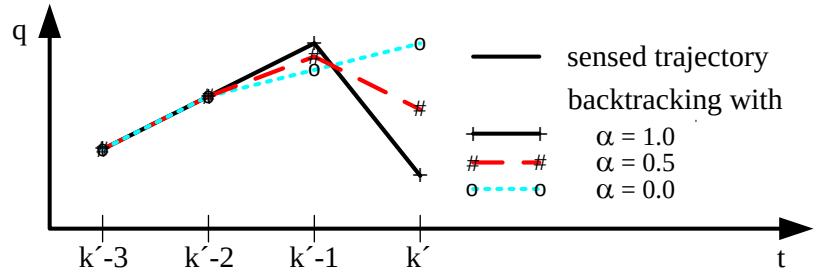

Figure 8. Effect of different values of α_{a_i} with backtracking of the acceleration.

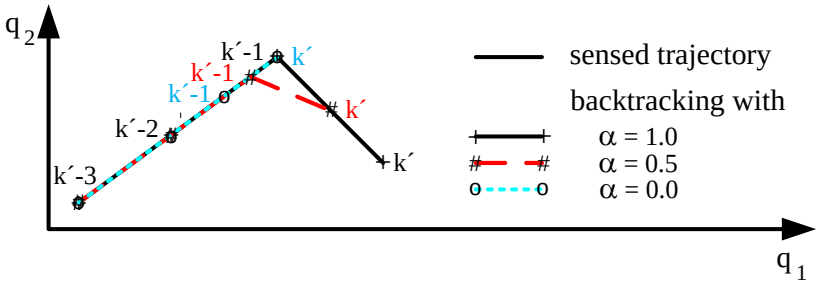

Figure 9. Resulting path after backtracking of the acceleration.

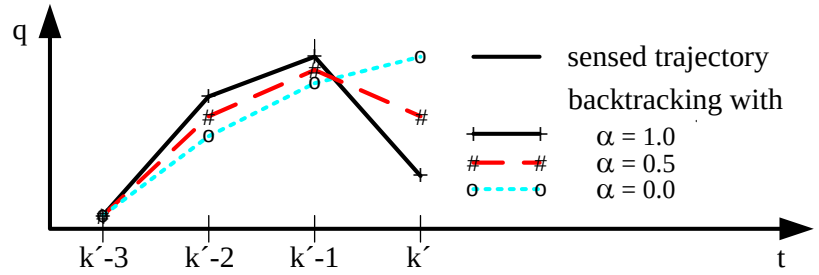

Figure 10. Effect of different values of α_{j_i} with backtracking of the jerk.

and

$$s_{*i}(k'') = s(k^-) + \frac{(q_{b*i}(k'') - q_{ci}(k^-)) \cdot (\underline{s}(k^-) + 1 - s(k^-))}{q_{di}(\underline{s}(k^-) + 1) - q_{ci}(k^-)} \tag{25}$$

instead of (11) and (13), and in

$$\mathbf{q}_c(k'') = \mathbf{q}_d(\underline{s}(k'')) + (s(k'') - \underline{s}(k'')) \cdot (\mathbf{q}_d(\underline{s}(k'') + 1) - \mathbf{q}_d(\underline{s}(k''))) \tag{26}$$

and

$$\mathbf{q}_c(k'') = \mathbf{q}_c(k^-) + \frac{(s(k'') - s(k^-)) \cdot (\mathbf{q}_d(\underline{s}(k^-) + 1) - \mathbf{q}_c(k^-))}{\underline{s}(k^-) + 1 - s(k^-)}, \tag{27}$$

instead of (15) and (16). (24) is computed with $\underline{s}_{*i}(k'') = \underline{s}(k''), \underline{s}(k'') - 1, \cdots, \underline{s}(k^-) + 1$, where

$s(k'')$ is initialized by the minimum of $s(k'')$ from a previous step of forward scaling or backtracking and $s(k''+1)$, if available. (25) is used if (24) does not result in a $\underline{s}_{*i}(k'') \leq s_{*i}(k'') < \min(\underline{s}_{*i}(k'')+1, s(k''))$. (27) is required instead of (26) if $s(k'') < \underline{s}(k^-)+1$.

When computing $\mathbf{q}_c(k'')$ as proposed, path accuracy is given, but the constraints of other axes are not always satisfied. Therefore, as with forward scaling, an iteration may be required within a backtracking step until all constraints are satisfied.

However, if a feasible solution is not reached within few iteration steps, convergence will be achieved by several backtracking steps with $\alpha_* = 0$, as $\alpha_* = 0$ definitely satisfies the corresponding constraint of $* = \{a_i, j_i\}$. A solution after ALI is found at the latest with $\underline{s}(k^-) = \underline{s}_{*i}(k' - \kappa_{min}) = \cdots = \underline{s}_{*i}(k')$, since then the interpolation concerns only a single interval of $\mathbf{q}_d$.

4. Extensions

In this section alternative realizations are presented that mitigate apparent disadvantages of the previously explained method. Among many possible heuristics for improving the convergence, two methods are explained in more detail. In Sect. 4.1 convergence problems with high-speed motion are reduced by a new backtracking approach. Sect. 4.2 estimates the prediction length κ_{max}. Sect. 4.3 then considers that a corner may not be executed properly or an (almost) closed loop will be cut short.

4.1. *Minimum Backtracking for Fast Target Motion*

When a modification of the sensed trajectory is required to reach feasibility, the motion is decelerated first. Then the tcp passes the critical position, e.g. a corner of the sensed path. Thereafter the robot is accelerated to a velocity that is higher as originally sensed, since otherwise the sensed trajectory will not be reached. This high-speed period ends by backtracking in order to avoid overshooting. The problem is that, for high-speed trajectories, backtracking easily decelerates too much such that the sensed trajectory is not reached anymore. This results in a permanent delay, which has not been considered in [35]. Fig. 15 in Sect. 6 gives an example where the normal backtracking algorithm results in a path-accurate but delayed motion.

Fig. 11 shows the details disregarding a jerk limit. For $k' = 20$, the approach of (17)-(18), which is now denoted as *normal backtracking* q_{b*}, modifies both, $q_d(k') = q_d(20)$ and $q_c(k'-1) = q_f(19)$ and thus decelerates significantly. If this happens repeatedly, then, the sensed trajectory will not be reached.

A less rigorous backtracking keeps the sensed position, i.e., $q_b(k') = q_d(k')$, if this is possible, and only modifies $q_c(k'-1)$. Three variants are displayed in Fig. 11 using $a_b(k') = \bar{a}$, $a_b(k') = 0$, and $a_b(k') = -\bar{a}$. With the smallest modification of $q_c(k'-1)$ this kind of backtracking results in minimum distance to $q_d(k'-1)$. Unfortunately, in this way the convergence is quite slow.

For a violation of an acceleration constraint $|a_{di}(k')| < \bar{a}_i$,

$$q_{bai}(k'-1) = (q_{di}(k') + q_{ci}(k'-2) \mp c_b \bar{a}_i)/2 \tag{28}$$

is set with a *convergence factor* of $-1 \leq c_b \leq 1$, where the sign of $\mp\bar{a}_i$ is contrary to the sign of $a_{di}(k')$.

Correspondingly,

$$q_{bji}(k'-1) = (q_{di}(k') + 3q_{ci}(k'-2) - q_{ci}(k'-3) \mp c_b \bar{j}_i)/3, \tag{29}$$

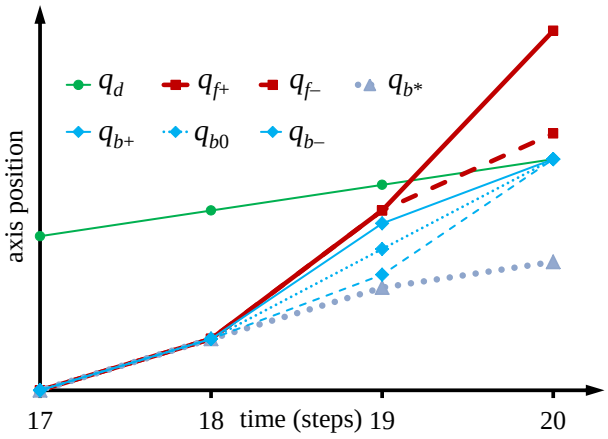

Figure 11. Reaching the sensed trajectory after a slowdown, disregarding jerk limits: q_d is the sensed trajectory, q_{f+} is the trajectory that is generated by forward scaling with a variant q_{f-} in which the sign of the acceleration is changed when exceeding the sensed trajectory. Nevertheless, the system will overshoot, such that backtracking is required. q_{b*} shows the normal backtracking of (17) and (18) whereas the other curves display alternative backtracking.

or

$$q_{bji}(k'-2) = (-q_{di}(k') + 3q_{ci}(k'-1) + q_{ci}(k'-3) \pm c_b\bar{j}_i)/3 \tag{30}$$

are computed whenever backtracking is required because a jerk constraint is not satisfied.

In (29) the sign of $\mp\bar{j}_i$ is contrary to the sign of $j_{di}(k')$ whereas in (30) the sign of $\pm\bar{j}_i$ is the same as that of $j_{di}(k')$. Only one of these equations shifts the corresponding position to the past. Therefore, (29) is used when $j_{di}(k')v_{ci}(k'-1) < 0$ since then the absolute value of the velocity $v_{ci}(k'-1)$ is reduced because of (29) with $\bar{j}_i < |j_{di}(k')|$. Otherwise it is increased which is accounted for by modifying $q_{ci}(k'-2)$.

Backtracking using (28)-(30) and $q_{b*i}(k') = q_{di}(k')$ is denoted as *minimum backtracking*. In contrast to the *normal backtracking* of (17)-(18) and (20)-(22), with minimum backtracking, $q_{b*i}(k'-1)$ and $q_{b*i}(k'-2)$ are not always in the respective interval $(q_{ci}(k'-2), q_{ci}(k'-1))$ or $(q_{ci}(k'-3), q_{ci}(k'-2))$. In these cases the normal backtracking algorithm is used.

Independently from this, the Arc Length Interpolation (24)-(27) is executed as before, resulting in path-accurate positions.

Other than in Fig. 11, the required deceleration is fundamental with high-speed and spans several sampling steps. Therefore a significant deceleration as that of the normal backtracking or of minimum backtracking with $c_b \ll 1$ speeds up the convergence, as long as the sensed trajectory is reached. Minimum backtracking with $c_b = 0$ turns out to be suitable whereas the normal backtracking may prevent from catching up the sensed trajectory.

4.2. *Adaptation of the Preview*

The required number of future time steps κ_{max} that have to be used for backtracking depends on the velocity and on how close the sensed trajectory is to the limits. Since the preview is limited, its maximum value $\bar{\kappa}_{max}$ is given by the leftmost block of Fig. 1.

But it is not yet investigated whether a violation of a constraint at a time step $k + \bar{\kappa}_{max}$ has to be solved at time step k. Therefore a required deceleration might be initiated too early. For example the orange plot in Fig. 16 in Sect. 6 seems to be shifted too much. This can be prevented by adapting the preview $\kappa_{max} \leq \bar{\kappa}_{max}$ as a function of the current and the future accelerations and

velocities.

Therefore, first the time $\kappa_{max}(k)$ is roughly estimated, which is needed to reach the current goal position $\mathbf{q}_d(k+\bar{\kappa}_{max})$, goal velocity $\mathbf{v}_d(k+\bar{\kappa}_{max})$, and goal acceleration $\mathbf{a}_d(k+\bar{\kappa}_{max})$ from $\mathbf{q}_d(k)$, $\mathbf{v}_d(k)$, and $\mathbf{a}_d(k)$, respectively. If the goal can be reached in less than $\bar{\kappa}_{max}$ sampling steps, the preview for trajectory generation is reduced from $\bar{\kappa}_{max}$ to $\kappa_{max}(k)$ sampling steps. Then $\mathbf{q}_d(k+\kappa)$ with $\kappa > \kappa_{max}(k)$ is not used for the computation of $\mathbf{q}_c(k)$ anymore.

This rule of thumb is not always correct because e.g. the time to reach $\mathbf{q}_d(k+\bar{\kappa}_{max}-2)$ from $\mathbf{q}_d(k)$ may be larger than that to reach $\mathbf{q}_d(k+\bar{\kappa}_{max})$. But in this case $\kappa_{max}(k-2)$ was probably big enough and thus initiated the appropriate modification of the trajectory.

In contrast to the preceding sections, for roughly estimating κ_{max} we assume phases with constant jerk. Unlike [8] however, we assume only up to three phases as we are close to the goal. At time step k the phases for axis i are

- $j_{i1} = \pm\bar{j}_i$ for $t = k, \cdots, k+t_{i1}(k)$,
- $a_{i2} = \pm\bar{a}_i$ for $t = k+t_{i1}(k), \cdots, k+t_{i1}(k)+t_{i2}(k)$,
- $j_{i3} = \pm\bar{j}_i$ for $t = k+t_{i1}(k)+t_{i2}(k), \cdots, k+t_{i1}(k)+t_{i2}(k)+t_{i3}(k)$.

With given initial and final conditions ($v_{di}(k)$, $a_{di}(k)$, $j_{di}(k)$, $v_{di}(k+\bar{\kappa}_{max})$, $a_{di}(k+\bar{\kappa}_{max})$, $j_{di}(k+\bar{\kappa}_{max})$) the phases result in 3 equations for $t_{i1}(k)$, $t_{i2}(k)$, and $t_{i3}(k)$.

$$
\begin{aligned}
a_{i2} &= a_{di}(k) + t_{i1} j_{i1} && (31)\\
a_{di}(k+\bar{\kappa}_{max}) &= a_{i2} + t_{i3} j_{i3} && (32)\\
v_{di}(k+\bar{\kappa}_{max}) &= v_{di}(k) + t_{i1} a_{di}(k) + t_{i1}(t_{i1}+1)/2\ j_{i1} \\
&\quad + t_{i2}(a_{di}(k) + t_{i1} j_{i1}) && (33)\\
&\quad + t_{i3}(a_{di}(k) + t_{i1} j_{i1}) + t_{i3}(t_{i3}+1)/2\ j_{i3}
\end{aligned}
$$

If no solution with $t_{i1} \geq 0$, $t_{i2} \geq 0$, and $t_{i3} \geq 0$ can be found, the second phase is probably not required. Then

- $j_{i1} = \pm\bar{j}_i$ for $t = k, \cdots, k+t_{i1}(k)$,
- $j_{i3} = \pm\bar{j}_i$ for $t = k+t_{i1}(k), \cdots, k+t_{i1}(k)+t_{i3}(k)$

is assumed, where (31)-(33) with $t_{i2} = 0$ give 3 equations for $t_{i1}(k)$, $a_{i2}(k)$ and $t_{i3}(k)$.

This system of equations is singular if $j_{i1} = j_{i3}$. Then only a single phase exists and either $a_{di}(k+\bar{\kappa}_{max})$ or $v_{di}(k+\bar{\kappa}_{max})$ can be reached by

- $j_{i3} = \pm\bar{j}_i$ for $t = k, \cdots, k+t_{i3}(k)$.

Otherwise, whenever no solution with $t_{i1} \geq 0$, $|a_{i2}| \leq \bar{a}_i$, and $t_{i3} \geq 0$ can be found with $t_{i2} = 0$,

$$
\begin{aligned}
t_{i1} &= 0 && (34)\\
t_{i2} &= |v_{di}(k+\bar{\kappa}_{max}) - v_{di}(k)|/\bar{a}_i && (35)\\
t_{i3} &= |a_{di}(k+\bar{\kappa}_{max}) - a_{di}(k)|/\bar{j}_i && (36)
\end{aligned}
$$

is used as a last resort.

Fig. 12 shows the different approaches for estimating the required prediction length.

Finally,

$$\kappa_{max}(k) = \max_i ceil(t_{i1}(k) + t_{i2}(k) + t_{i3}(k)) + 1 \quad (37)$$

is computed where $ceil(\cdot)$ denotes the smallest integer that is not less than the argument. This is required since, strictly speaking, the jerk cannot change within a sampling step. Therefore in

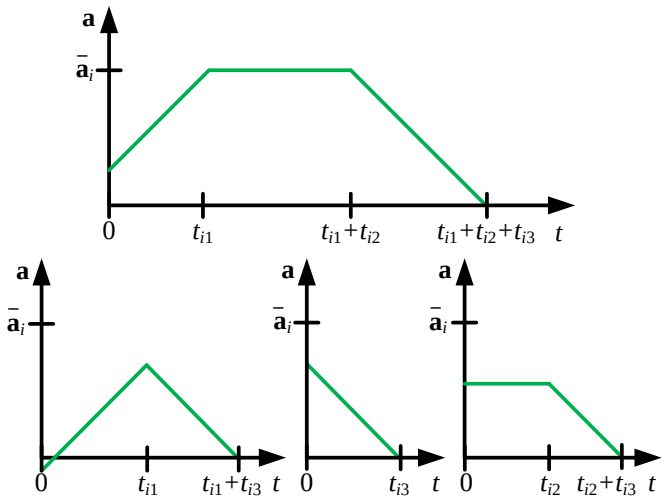

Figure 12. Approaches with one, two, or three phases for estimating the time required to reach the goal position, depending on the initial and the final values of the position, the velocity, and the acceleration.

reality, during the phases 1 and 3, a smaller value $|j_{i*}| \leq \bar{j}_i$ is applied for a larger time $ceil(t_{i*}) \geq t_{i*}$.

$\kappa_{max}(k)$ is cut back to the range

$$\underline{\kappa}_{max} \leq \kappa_{max}(k) \leq \bar{\kappa}_{max} \tag{38}$$

with $\underline{\kappa}_{max} = 5$ and $\bar{\kappa}_{max} = 21$.

Though this assessment implies some heuristics, Sect. 6 proves that the procedure is useful.

4.3. *Execution of Corners and Closed Loops in the Path*

If the sensed path shows closed loops (as in Fig. 17 in Sect. 6), it is not guaranteed that the loops are executed. Instead, if the trajectory was substantially decelerated before, the algorithm might try to skip the loop and continue with the following segment of the path. The same applies when a future position is close to a another one, without representing a proper loop.

Similarly, a corner in the path may be blended. Though ALI from Sect. 3 inhibits sampled positions distant from the sensed path, it is not assured that a corner of the sensed path is represented by a sampling step of the commanded trajectory, see Fig. 7.

Cutting short of loops or corners can be inhibited by starting the iteration for $s(k')$ close to $s(k'-1)$, e.g. at $s_{start} = \min(k', s(k'-1) + \overline{\Delta s})$ with $\overline{\Delta s} = 5$, instead of at k'. This implies that $\mathbf{q}_d(s_{start})$ instead of $\mathbf{q}_d(k')$ is checked for feasibility and taken as the first attempt of (11) or (24). Similarly, during the iteration, $s(k')$ will be replaced by $s(k'-1)+\overline{\Delta s}$ whenever the latter is smaller, e.g. when $s(k'-1)$ has been reduced by backtracking after $s(k')$.

As a result all sampled positions of the commanded trajectory $\mathbf{q}_c(k')$ have a temporal distance of at most $\overline{\Delta s}$ steps of $\mathbf{q}_d$. Thus, as a rule of thumb, the position error is not more than $\mathbf{q}_d(k' + \overline{\Delta s}) - \mathbf{q}_d(k')$, i.e., the path of the sensed trajectory executed within $\overline{\Delta s}$. This limits the distance of a sensed corner from the commanded path. Due to the kinematic constraints the distance of sampled commanded positions to a corner is already largely restricted by the reduced velocity (see Sect. 6). However, for pointed corners a small $\overline{\Delta s}$ will be more restrictive than that.

A closed loop is executed without fail if it lasts at least $\overline{\Delta s}$ sampling steps of the sensed trajectory. There is no cut short between two positions whose temporal difference in the sensed trajectory is more than $\overline{\Delta s}$.

The drawback is that the catch up after a significant deceleration may be delayed in the case

that the iteration for $s(k')$ begins too close to $s(k'-1)$. Sect. 6 shows the effect of different values of $\overline{\Delta s}$.

5. Discussion

The presented method is different with respect to most existing algorithms insofar as no continuous or differentiable sensed trajectory is assumed, which means that e.g. the acceleration is computed at each time step from sampled positions. This results in features of the method that will be outlined in this section.

5.1. *Performance*

First of all, the presented method considers the discrete-time constraints (1)-(3) exactly, i.e., it accounts for each sampled position of the sensed desired trajectory. Furthermore, there is no linearization between sampled positions of the sensed and the commanded trajectory, as e.g. between $\mathbf{q}_c(k-1)$ and $\mathbf{q}_d(k)$ in [40]. However, in general the iteration does not result in an optimal trajectory, meaning that the constraints are not always fully utilized. But they are definitely not violated.

This does not only apply as long as the iteration is successful, i.e., as long as a path-accurate trajectory is found. The computed trajectory is still feasible whenever Direct Scaling is used as a fallback position.

But there are desired trajectories for which no feasible modification is possible at all. This may happen if the robot has accelerated until the maximum velocity is reached. But then the robot continues to accelerate because the constraints on the jerk do not allow zero acceleration in the next step. This is called a *forbidden point* in [6]. In this case for a short time a slight violation of the maximum speed is unavoidable. Alternatively, the maximum jerk may be exceeded.

5.2. *Stability*

The robot's stability is not affected when modifying a sensed trajectory to a feasible one, since the modification of the desired trajectory does not change the servo control loop. On the other hand, it has been shown in [1] that the two feedback signals of the position and the sensor data (see Fig. 1) cancel each other out so that no other feedback loop is present. Instead, $\mathbf{q}_d$ represents a trajectory that is given by the task definition and not by the robot pose. In this way, stability would only be at risk if the trajectory generation itself would be unstable. This, however, is BIBO (bounded input - bounded output) stable with respect to the position since,

- for a path accurate trajectory, each component of the commanded trajectory is bounded by the extremal values of this component of the sensed trajectory,
- otherwise, the difference with respect to the sensed trajectory is bounded by Direct Scaling. The worst case is an overshooting from maximum velocity $\bar{v}_i$ and maximum acceleration $\bar{a}_i$, that needs at most $\kappa_j = \max_i ceil(2\bar{a}_i/\bar{j}_i)$ sampling steps to attain the maximum deceleration and then at most $\kappa_a = \max_i ceil(\bar{v}_i/\bar{a}_i)$ sampling steps to come to a stop. According to (A3) and (A1) this results in a upper bound for the deviation of $\delta_i = \kappa_j\bar{v}_i + \kappa_j(\kappa_j+1)/2\ \bar{a}_i + \kappa_j(\kappa_j+1)((\kappa_j+2)/6\ \bar{j}_i + \kappa_a\bar{v}_i + \kappa_a(\kappa_a+1)/2\ \bar{a}_i$.

Thus, if $|q_{di}(\kappa) - q_{di}(0)| < \delta\ \ \forall\kappa = 0, \cdots, k$, then $|q_{ci}(\kappa) - q_{di}(0)| < \delta + \delta_i\ \ \forall\kappa = 0, \cdots, k$.

In addition, since the trajectory is feasible, the derivatives are bounded by the constraints. But this does not exclude that the resulting velocity $|\mathbf{v}_c(k)| = |\mathbf{q}_c(k) - \mathbf{q}_c(k-1)|$ of the commanded trajectory may temporarily be higher than the input velocity $|\mathbf{q}_d(k) - \mathbf{q}_d(k-1)|$ of the sensed trajectory. It is only assured that $|\mathbf{v}_c(k)| \leq |\mathbf{q}_d(k) - \mathbf{q}_c(k-1)|$.

5.3. *Convergence*

During the modification of the trajectory, it is slowed down in each step of path accurate forward scaling or backtracking because with (17)-(23) and $s(k) \leq k$ or with (28)-(30) the positions are shifted towards positions at previous time steps. This inhibits an oscillation of $\mathbf{v}_c(k)$ around $\mathbf{v}_d(k)$. But it does not guarantee convergence. Convergence to a path-accurate trajectory cannot be assured since, as noted, a path-accurate solution does not always exist.

Beyond that, it is possible that a modification is too big. This is due to (17)-(18), (20)-(22), or to the method of Sect. 4.1, which heuristically reduce the problem of computing two or three positions to a single α or to $\mathbf{q}_b(k') = \mathbf{q}_d(k')$ and (28)-(30). These approaches have been selected for the online computation in order to save computing power. Consequently the performance may be sub-optimal, more than ever since, once the arc length is reduced in iteration step l to $s^l(k) \leq k$, it is not intended to compute an $s^{l+1}(k) > s^l(k)$ in a further iteration step $l+1$. Therefore the convergence with the minimum backtracking of Sect. 4.1 is significantly better than with the method of [35].

6. Experiments

The experimental setup is similar to that in [35]. Similarly, a KUKA KR16 robot is controlled by RSI Ethernet to overlay a horizontal and a vertical motion, where the vertical motion is stopped by a (simulated) sensor at a minimum distance to the table (see Fig. 13). The table pose is detected in advance, as with a distance sensor. In this way, the sensed path can be executed path-accurately. However, the velocity is reduced temporarily before the corner in order to account for the constraints of Table 1.

Figs. 14 to 16 repeat the fastest experiment from [35] with twice as much speed, i.e., 200 mm/s.

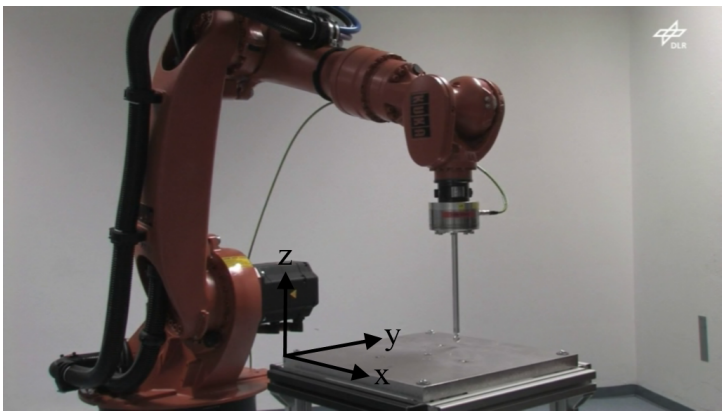

Figure 13. KUKA robot at the minimum distance to the table.

Table 1. Assumed limits of the individual axes (before filtering). The units for velocity, acceleration, and jerk are rad/(sampling step), rad/(sampling step)^2, and rad/(sampling step)^3, respectively.

i	$\bar{v}_i$	$\bar{a}_i$	$\bar{j}_i$	$\bar{j}'_i$
1	$14 \cdot 10^{-3}$	$74 \cdot 10^{-6}$	$60 \cdot 10^{-6}$	$24 \cdot 10^{-6}$
2	$14 \cdot 10^{-3}$	$37 \cdot 10^{-6}$	$30 \cdot 10^{-6}$	$12 \cdot 10^{-6}$
3	$14 \cdot 10^{-3}$	$85 \cdot 10^{-6}$	$69 \cdot 10^{-6}$	$28 \cdot 10^{-6}$
4	$30 \cdot 10^{-3}$	$250 \cdot 10^{-6}$	$204 \cdot 10^{-6}$	$82 \cdot 10^{-6}$
5	$30 \cdot 10^{-3}$	$252 \cdot 10^{-6}$	$206 \cdot 10^{-6}$	$82 \cdot 10^{-6}$
6	$56 \cdot 10^{-3}$	$450 \cdot 10^{-6}$	$368 \cdot 10^{-6}$	$147 \cdot 10^{-6}$

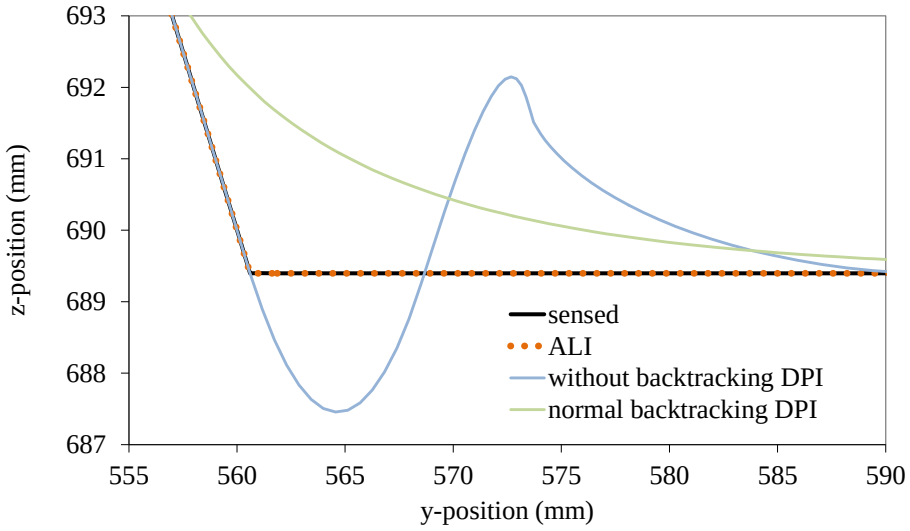

Figure 14. Sensed and generated paths in the y-z-plane (twice as fast as in [35]).

Regarding the path accuracy, Fig. 14 shows that the method without backtracking exhibits overshooting whereas the other one with DPI blends the corner. In contrast, path accuracy is not an issue for the versions of backtracking with ALI.

Regarding the plots over time, the normal backtracking with $\bar{\mathbf{j}}$ (orange dotted curve) copes with the corner, but in contrast to the reduced speed of [35], with 200 mm/s there is a delay such that the sensed trajectory is not reached before the end of the horizontal motion (Fig. 15). Using the more restrictive jerk limit $\bar{\mathbf{j}}'$, the normal backtracking (orange dashed curve) fails. In contrast, the experiments with the minimum backtracking of Sect. 4.1 (green and red curves) easily catch up, even with reduced limits (Fig. 15).

Figs. 15 and 16 also show that with minimum backtracking the delay is less than with normal backtracking. An even smaller delay is reached with the adapted preview of Sect. 4.2 (red curves). Without adapting κ_{max}, a feasible trajectory is also reached, but not the one with the smallest delay.

Figs. 17 and 18 show the effect of s_{start} (Sect. 4.3) with a modified path that includes a closed loop of 10 sampling steps. With $\overline{\Delta_s} \leq 5$ the loop is executed, whereas with $\overline{\Delta_s} = 10$ (or the procedure without the approach of Sect. 4.3) the loop is skipped (Fig. 17). This is because the commanded trajectory is delayed by 15 sampling steps when passing the corner.

Fig. 18 further shows that it is easy to catch up the sensed trajectory after a cut short of the loop. Otherwise a further slowdown is required before the loop. Thereafter, the smaller $\overline{\Delta_s}$, the later the sensed trajectory is reached. With $\overline{\Delta_s} = 2$ the slope is only twice as high as that of the sensed trajectory, with $\overline{\Delta_s} = 1$ the delay cannot be reduced at all. $\overline{\Delta_s} = 5$ seems to be a reasonable value. It does not result in any changes when applied to the experiment of Figs. 14 to 16.

The reader is encouraged to provide to the author additional test data, i.e., trajectories, preferably including predictions from each time step, and the given limits. The resulting trajectories will be returned promptly. The data format is given at http://rmc.dlr.de/rm/en/staff/friedrich.lange/trajectory-generation.

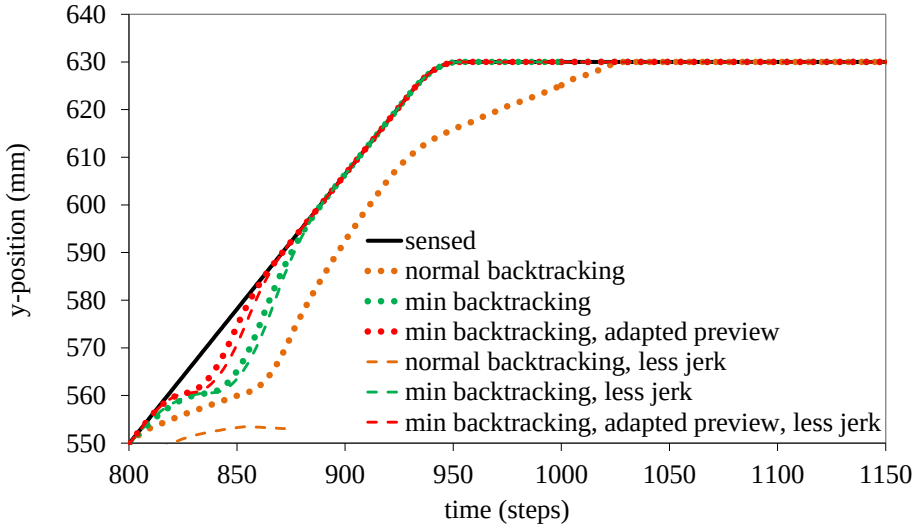

Figure 15. y component of the sensed and the generated trajectories.

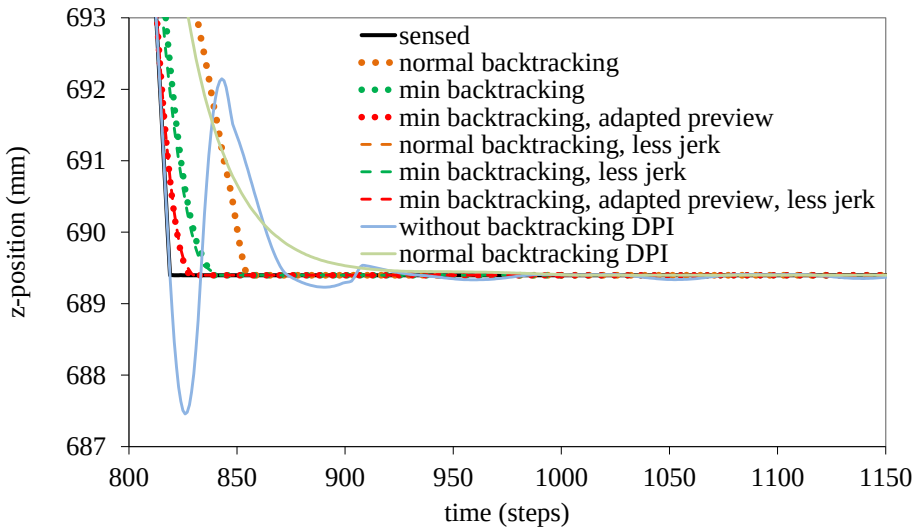

Figure 16. z component of the sensed and the generated trajectories.

7. Conclusion

The paper has presented an easy technique for the implementation of robot motion control. It satisfies all constraints of an industrial robot, thus enabling the execution of sensor-based modifications of a programmed trajectory.

While the Direct Position Interpolation (DPI) of [40] computes a position $\mathbf{q}_c(k)$ on the straight line between $\mathbf{q}_c(k-1)$ and $\mathbf{q}_d(k)$, Arc Length Interpolation (ALI) as presented in Sect. 3 selects the appropriate segment of the sensed path and interpolates between $\mathbf{q}_d(\underline{s}(k))$ and $\mathbf{q}_d(\underline{s}(k)+1)$ or between $\mathbf{q}_c(k-1)$ and $\mathbf{q}_d(\underline{s}(k-1)+1)$. This means that the commands $\mathbf{q}_c(k)$ are computed as $\mathbf{q}_c(k) = \mathbf{q}_d(s(k))$, where the latter expression stands for an interpolation if the argument of $\mathbf{q}_d(\cdot)$ is

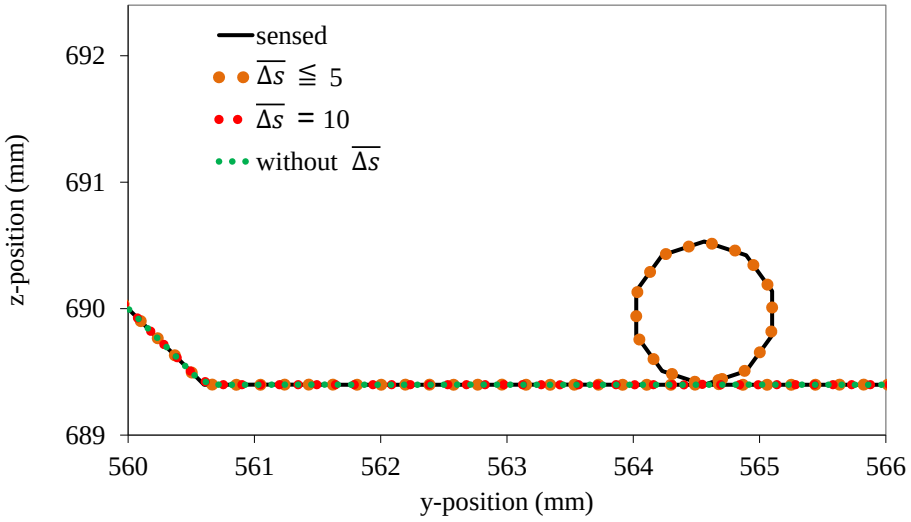

Figure 17. Sensed and generated paths with loop in the y-z-plane.

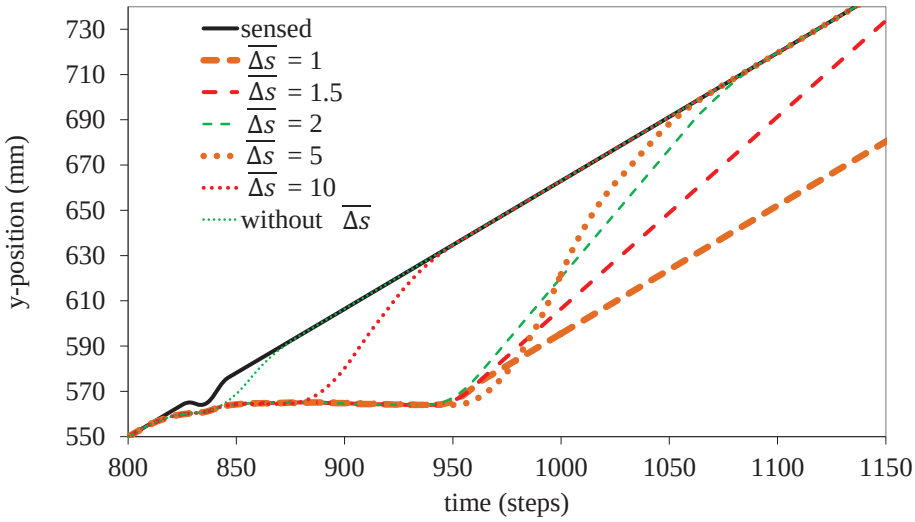

Figure 18. y component of the sensed and the generated trajectories with loop.

not integral. $s(k)$ is the time at which the sensed trajectory $\mathbf{q}_d$ has passed the commanded position $\mathbf{q}_c(k)$ that satisfies all constraints.

ALI reduces blending of corners that may arise with DPI. Beyond that, the method of Sect. 4.3 inhibits undesired shortcuts. Thus, the exact shape of the sensed path near $\mathbf{q}_c(k)$ in ensured.

$s(k)$ is not defined if $\mathbf{q}_c(k)$ is computed by Direct Scaling (DS), since then $\mathbf{q}_c(k)$ is not on the sensed path $\mathbf{q}_d(\cdot)$. The use of ALI is therefore not possible, which is not a disadvantage since the path is left anyway.

The overall trajectory is generated by the iterative approach of Sect. 2. The procedure always converges to a feasible trajectory, at the latest with Direct Scaling (DS) which is possible with-

out prerequisites. The convergence to a path-accurate trajectory is significantly improved by the procedures of Sects. 4.1 and 4.2.

Thus the main contribution of the paper with respect to the previous work in [35] lies in the extensions described in Sect. 4. These extensions ensure fast convergence and exclude premature reactions or undesired shortcuts, even for difficult trajectories.

Appendix A. Prediction from Computed Differences

The sampling time is set to $T_0 = 1$. This means that $T_0\dot{\mathbf{q}}$, $T_0^2\ddot{\mathbf{q}}$, and $T_0^3\dddot{\mathbf{q}}$ are replaced by $\mathbf{v}$, $\mathbf{a}$, and $\mathbf{j}$. This implies that the units of $\mathbf{q}$, $\mathbf{v}$, $\mathbf{a}$, and $\mathbf{j}$ are all identical.

Moreover, as in [40], all derivatives are computed by backward differences, e.g. $\mathbf{a}(k) = \mathbf{v}(k) - \mathbf{v}(k-1) = \mathbf{q}(k) - 2\mathbf{q}(k-1) + \mathbf{q}(k-2)$.

With $\mathbf{a}(k+1) = \cdots = \mathbf{a}(k+i)$ this results in

$$\mathbf{q}(k+i) = \mathbf{q}(k) + i\mathbf{v}(k) + i(i+1)/2\ \mathbf{a}(k+i) \tag{A1}$$

and

$$\mathbf{v}(k+i) = \mathbf{v}(k) + i\mathbf{a}(k+i). \tag{A2}$$

Similar to the acceleration, the jerk is defined by $\mathbf{j}(k) = \mathbf{a}(k) - \mathbf{a}(k-1) = \mathbf{q}(k) - 3\mathbf{q}(k-1) + 3\mathbf{q}(k-2) - \mathbf{q}(k-3)$. With $\mathbf{j}(k+1) = \cdots = \mathbf{j}(k+i)$ this results in

$$\begin{aligned}
\mathbf{q}(k+i) &= \mathbf{q}(k) + i\mathbf{v}(k) + i(i+1)/2\ \mathbf{a}(k) + i(i+1)(i+2)/6\ \mathbf{j}(k+i) && \text{(A3)}\\
\mathbf{v}(k+i) &= \mathbf{v}(k) + i\mathbf{a}(k) + i(i+1)/2\ \mathbf{j}(k+i) && \text{(A4)}\\
\mathbf{a}(k+i) &= \mathbf{a}(k) + i\mathbf{j}(k+i). && \text{(A5)}
\end{aligned}$$

For a single sampling step (A1) or (A3) correspond to

$$\begin{aligned}
\mathbf{q}(k+1) &= \mathbf{q}(k) + \mathbf{v}(k+1)\\
&= \mathbf{q}(k) + \mathbf{v}(k) + \mathbf{a}(k+1)\\
&= \mathbf{q}(k) + \mathbf{v}(k) + \mathbf{a}(k) + \mathbf{j}(k+1).
\end{aligned} \tag{A6}$$

Note that (A1), (A3), (A4), and (A6) differ from the classical Taylor expansion

$$\mathbf{q}(k+i) = \mathbf{q}(k) + i\dot{\mathbf{q}}(k) + i^2/2\ \ddot{\mathbf{q}}(k) + i^3/6\ \dddot{\mathbf{q}}(k) + \mathcal{O}(\mathbf{q}^{(4)}(k)). \tag{A7}$$

However, a comparison of both approaches shows that for functions of type $\mathbf{q}(k) = \mathbf{a} + \mathbf{b}k + \mathbf{c}k^2 + \mathbf{d}k^3$ the equations (A3) and (A7) are identical and the constrained differences result in $\mathbf{v}(k) = \dot{\mathbf{q}}(k - \mu_v)$ with $\mu_v \approx 0.5$, $\mathbf{a}(k) = \ddot{\mathbf{q}}(k-1)$, and $\mathbf{j} = \dddot{\mathbf{q}}$.

Beyond that, according to the mean value theorem, for any function $\mathbf{q}(k)$ which is three times differentiable between the sampled positions, the differences are $\mathbf{v}(k) = \dot{\mathbf{q}}(k-\mu_v)$, $\mathbf{a}(k) = \ddot{\mathbf{q}}(k-\mu_a)$, and $\mathbf{j}(k) = \dddot{\mathbf{q}}\,(k - \mu_j)$ with $0 \le \mu_v \le 1$, $0 \le \mu_a \le 2$, and $0 \le \mu_j \le 3$. This means that whenever $\mathbf{v}$, $\mathbf{a}$, and $\mathbf{j}$ satisfy the constraints for all time steps, the trajectory is also feasible with respect to $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, and $\dddot{\mathbf{q}}$, besides violations within the sampling intervals, which cannot be considered by the time-discrete approach. Thus the reduction of the limits $\bar{\mathbf{a}}$ and $\bar{\mathbf{j}}$, which was proposed in [35] as a precaution, is not required.

References

[1] F. Lange, W. Bertleff, and M. Suppa. Force and trajectory control of industrial robots in stiff contact. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2912–2919, Karlsruhe, Germany, May 2013.

[2] J. Schultz and T. D. Murphey. Real-time trajectory generation for a planar crane using discrete mechanics. In *Workshop on "Real-time Motion Generation & Control - Constraint-based Robot Programming" at the 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, Sep 2014. `"http://cs.stanford.edu/people/tkr/iros2014/proceedings.html"`.

[3] Y. Suzuki, K. Koyama, A. Ming, and M. Shimojo. Grasping strategy for moving object using net-structure proximity sensor and vision sensor. In *Proc. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1403–1409, Seattle, Washington, USA, May 2015.

[4] F. Lange, J. Werner, J. Scharrer, and G. Hirzinger. Assembling wheels to continuously conveyed car bodies using a standard industrial robot. In *Proc. 2010 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3863–3869, Anchorage, AK, USA, May 2010.

[5] S. Macfarlane and E. A. Croft. Jerk-bounded manipulators trajectory planning: Design for real-time applications. *IEEE Trans. on Robotics*, 19(1):42–52, Feb 2003.

[6] J. Mattmüller and D. Gisler. Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path. *Int. J. Adv. Manuf. Technol.*, 45:1007–1016, 2009.

[7] F. Ghilardelli, C. Guarino Lo Bianco, and M. Locatelli. Smart changes of the end-effector orientation for the automatic handling of singular configurations. *IEEE/ASME Trans. on Mechatronics*, 21(4):2154–2164, Aug 2016. DOI: 10.1109/TMECH.2015.2506679.

[8] T. Kröger and F. M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Trans. on Robotics*, 26(1):94–111, Feb 2010.

[9] M. H. Ghasemi, N. Kashiri, and M. Dardel. Near time-optimal control of redundant manipulators along a specified path with jerk constraint. *Advanced Robotics*, 25:2319–2339, 2011.

[10] F. Lange and M. Suppa. Trajectory generation for immediate path-accurate stopping of industrial robots. In *Proc. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2021–2026, Seattle, WA, USA, May 2015.

[11] L. van Aken and H. van Brussel. On-line robot trajectory control in joint coordinates by means of imposed acceleration profiles. *Robotica*, 6(3):185–195, July 1988.

[12] Y. Bestaoui. On-line motion generation with velocity and acceleration constraints. *Robotics and Autonomous Systems*, 5:279–288, 3 1989.

[13] T. Kröger. The reflexxes motion libraries: An introduction to instantaneous trajectory generation. Tutorial Mo-T-19 at 2013 IEEE Int. Conf. on Robotics and Automation (ICRA), May 2013.

[14] Reflexxes. `http://www.reflexxes.ws/`, last visited 2015.

[15] T. Kröger. Online trajectory generation: Straight-line trajectories. *IEEE Trans. on Robotics*, 27(5):1010–1016, Oct 2011.

[16] S. A. Bazaz and B. Tondu. Minimum-time on-line joint trajectory generator based on low order spline method for industrial manipulators. *Robotics and Autonomous Systems*, 29(4):257–268, 1999.

[17] K. Erkorkmaz and Y. Altintas. High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. *Int. J. of Machine Tools and Manufacture*, 41(9):1323–1345, July 2001.

[18] R. Haschke, E. Weitnauer, and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3248–3253, Nice, France, Sep 2008.

[19] L. Biagiotti and C. Melchiorri. Online trajectory planning and filtering for robotic applicatons via B-spline smoothing fiters. In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5668–5673, Tokyo, Japan, Nov. 2013.

[20] R. L. Williams II. Simplified robotics joint-space trajectory generation with a via point using a single polynomial. *Journal of Robotics*, 2013. http://dx.doi.org/10.1155/2013/735958.

[21] L. Yang, D. Song, J. Xiao, J. Han, L. Yang, and Y. Cao. Generation of dynamically feasible and collision free trajectory by applying six-order bezier curve and local optimal reshaping. In *Proc. 2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 643–648, Hamburg, Germany, Sep/Oct 2015.

[22] M. M. Ghazaei Ardakani. *Topics in Trajectory Generation for Robots.* PhD Thesis, Lund University,

Sweden, Department of Automatic Control, 2015.

[23] K. Hauser. Fast dynamic optimization of robot paths under actuator limits and frictional contact. In *Proc. 2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2990–2996, Hong Kong, China, May/June 2014.

[24] O. Dahl and L. Nielsen. Torque limited path following by on-line trajectory time scaling. *IEEE Trans. on Robotics and Automation*, 6(5):554–561, Oct 1990.

[25] Z. Shiller and H.-H Lu. Computation of path constrained time optimal motions with dynamic singularities. *ASME Journal of Dynamic Systems, Measurements, and Control*, 114:34–40, 1 1992.

[26] C. Guarino Lo Bianco and O. Gerelli. Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints. *IEEE Trans. on Robotics*, 27(6):1144–1152, Dec 2011.

[27] C. Guarino Lo Bianco and F. Ghilardelli. Techniques to preserve the stability of a trajectory scaling algorithm. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 862–868, Karlsruhe, Germany, May 2013.

[28] R. Gill, D. Kulić, and C. Nielsen. Robust path following for robot manipulators. In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3412–3418, Tokyo, Japan, Nov. 2013.

[29] A. Amthor, A. Lorenz, S. Zschaeck, and C. Ament. Fourth order motion profile planning for high precision applications. In *Proc. 2010 IASTED Int. Conf. on Robotics*, volume 1, pages 55–61, Phuket, Thailand, Nov 2010.

[30] F. Debrouwere, W. Van Loock, G. Pipeleers, Q. T. Dinh, M. Diehl, J. De Schutter, and J. Swevers. Time-optimal path following for robots with convex-concave constraints using sequential convex programming. *IEEE Trans. on Robotics*, 29(6):1485–1495, Dec 2013.

[31] R. Katzschmann, T. Kröger, T. Asfour, and O. Khatib. Towards online trajectory generation considering robot dynamics and torque limits. In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5644–5651, Tokyo, Japan, Nov. 2013.

[32] C. Guarino Lo Bianco and F. Ghilardelli. A discrete-time filter for the generation of signals with asymmetric and variable bounds on velocity, acceleration, and jerks. *IEEE Trans. on Industrial Electronics*, 61(8):4115–4125, Aug 2014.

[33] Q. Pham. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Trans. on Robotics*, 30(6):1533–1540, Dec. 2014.

[34] A. K. Singh and K. M. Krishna. A class of non-linear time scaling functions for smooth time optimal control along specified paths. In *Proc. 2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5809–5816, Hamburg, Germany, Sep/Oct 2015.

[35] F. Lange and A. Albu-Schäffer. Path-accurate online trajectory generation for jerk-limited industrial robots. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):82–89, 2016. Presented at the *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

[36] S. Pchelkin, A. Shiriaev, A. Robertsson, and L. Freidovich. Integrated time-optimal trajectory planning and control design for industrial robot manipulator. In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2521–2526, Tokyo, Japan, Nov. 2013.

[37] F. Flacco and A. De Luca. Optimal redundancy resolution with task scaling under hard bounds in the robot joint space. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3954–3960, Karlsruhe, Germany, May 2013.

[38] F. Flacco and A. De Luca. Fast redundancy resolution for high-dimensional robots executing prioritized tasks under hard bounds in the joint space. In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2500–2506, Tokyo, Japan, Nov. 2013.

[39] T. Kunz and M. Stilman. Probalistically complete kinodynamic planning for robot manipulators with acceleration limits. In *Proc. 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3713–3819, Chicago, IL, USA, Sep 2014.

[40] F. Lange and M. Suppa. Predictive path-accurate scaling of a sensor-based defined trajectory. In *Proc. 2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 754–759, Hong Kong, China, May/June 2014.